UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/771,678	02/04/2004	Gerard M. Col	CNTR.2094	4814

23669 7590 04/10/2006

HUFFMAN LAW GROUP, P.C.
1832 N. CASCADE AVE.
COLORADO SPRINGS, CO 80907-7449

EXAMINER
MOLL, JESSE R

ART UNIT	PAPER NUMBER
2181	

DATE MAILED: 04/10/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

e

Office Action Summary	Application No. 10/771,678	Applicant(s) COL, GERARD M.	
	Examiner Jesse R. Moll	Art Unit 2181	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 13 July 2005.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-43 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-40 and 43 is/are rejected.

7)☒ Claim(s) 41 and 42 is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 04 February 2004 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Supervisory
FRITZ FLEMING
PRIMARY EXAMINER
GROUP 2100
AU 2181
4/2/2006

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 13 July 2005.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-43 have been examined.

Acknowledgment of papers filed: oath, specification, drawings, and IDS on February 4, 2004; IDS on September 8, 2004; and IDS on July 13, 2005. The papers filed have been placed on record.

Claim Objections

2. Claim 14 is objected to because it is unclear whether the limitation "an early register file" is different from "a storage element" as claimed in claim 1. Examiner requests that if the two limitations are intended to claim the same object, "an early register file" be replaced with "said storage element".

3. Regarding claim 29, Examiner requests that "further comprising: early execution logic, coupled to said logic," read "wherein said early execution logic is coupled to said logic, and is" because the limitation "early execution unit" is already recited in parent claim 22.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 21 and 43 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claims 21 and 43 recite the limitation “a computer data signal embodied in a transmission medium”. A computer data signal embodied in a transmission medium does not fall under any of the statutory classes. First, a claimed signal is clearly not a "process" under Sec. 101 because it is not a series of steps. The other three Sec. 101 classes of machine, compositions of matter and manufactures "relate to structural entities and can be grouped as `product' claims in order to contrast them with process claims." 1 D. Chisum, Patents Sec. 1.02 (1994). The three product classes have traditionally required physical structure or material.

Examiner suggests (and assumes for the purpose of examination) that the limitation “A computer data signal embodied in a transmission medium” read “A computer program embodied on a computer readable medium”. The term computer usable (e.g., readable) medium as defined in Applicant’s specification (page 43, lines 7-10) does not include non-tangible signals.

Claim Rejections - 35 USC § 112

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claims 8, 13, 20, 21 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

7. Claim 8 recites the limitation "commonly". It is unclear how the term "commonly " limits the claim. For the purpose of examination, Examiner will not give patentable weight to this limitation.

8. Claim 13 recites the limitation "said early execution logic is within a stage of the pipeline microprocessor immediately following a stage of the microprocessor including an architected register file of the microprocessor". It is unclear how two physical objects can be in stages that follow one another. Stage can be interpreted as either a distinct physical part of something, or as a distinct time period. If the term "stage" is interpreted as a physical part, it is unclear how one of the stages is "immediately following". If the term "stage" is interpreted as a distinct time period, it is unclear how physical objects exist in a period of time. Further, according to Applicant's specification, the architected register file is update via result bus 158 in the W stage (see paragraph 35, lines 7-8). It is unclear how the architected register file 134 is contained within one stage because it is accessed more than once by the same instruction. For the purpose of examination, Examiner assumes the limitation read "said early results are generated immediately following operands corresponding to said early results being read from the architected register file".

9. Claim 20 recites the limitation "wherein a computer program product... causes the apparatus". It is unclear how a computer program on a medium can cause a

processor. Manufacturing a processor is done by a physical machine to print the circuitry. Any code for the processor is merely used by the machine. Further, the term “causes” is unclear because it is not clear if the limitation claims the code creating the processor, or the code causing the processor to do something. For the purpose of examination, Examiner assumes the code causes the processor to operate.

10. Claims 21 and 43 are rejected because of similar reasons as claim 20. The term “provides” is equally as unclear as “causes”. For purpose of examination, Examiner assumes the code provides the apparatus with data.

Claim Rejections - 35 USC § 102

11. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

12. Claims 1, 2, 4, 8-40, and 43 are rejected under 35 U.S.C. 102(b) as being anticipated by Denman (U.S. Patent No. 5,493,669).

13. Regarding claim 1, Denman discloses an apparatus for generating early status flags to enable early execution (execution before data is available from architected registers) of a conditional instruction (see col. 6, lines 44-62;

Note that according to The American Heritage® Dictionary of the English Language, Fourth Edition the definition of the word conditional is "Imposing, depending *on, or containing a condition." It also states that the definition of the word condition is "A mode or state of being". Using these definitions, every instruction in a processor imposes or depends on the state of the processor.)*

in a pipeline microprocessor having architected status flags (Architectural Registers; see col. 3, lines 22-25;

Note that any register stores a status and therefore can be considered a status flag.),

the apparatus comprising: a storage element (Rename Buffer 14; see fig. 1), for accumulating early status flags (Results) corresponding to the architected status flags (see col. 4, lines 26-28); and logic (Sequencer), coupled to said storage element (see col. 4, lines 29-30), configured to update said early status flags in said storage element in response to early results (see col. 4, lines 26-27;

Note that the results are considered early results because the results are obtained before the instruction is completed and written back to the architectural state.)

of instructions preceding the conditional instruction (all instructions with operands needed by the conditional instruction), wherein said logic invalidates said early status flags (see col. 10, lines 14-17) if at least one of said early results of said instructions that modify said early status flags is invalid (see col. 10, lines 11-14;

Note that if a branch is mispredicted, the instructions are invalid because they should not have been updated.),

thereby enabling, if said early status flags are valid (see col. 6, lines 52-56), execution of the conditional instruction based on said early status flags (see col. 6, lines

49-52) prior to the microprocessor updating the architected status flags (see col. 4, last 3 lines) in response to final results generated for said preceding instructions.

Note that if the results must be generated in order for the architected status flags to be updated. Therefore, the updating occurs in response to the final results.

14. Regarding claim 2, Denman discloses the apparatus of claim 1, further comprising: a first stage of the microprocessor pipeline, wherein said logic generates said early status flags; a second stage of the microprocessor pipeline, for updating the architected status flags; wherein said first stage is earlier in the microprocessor pipeline than said second stage (see col. 6, lines 45-62; col. 4, lines 65-68).

Note that if the entry exists in the rename buffer, the early status flags must be updated before the architected status flags because entries are invalidated when the architected status flags are updated.

15. Regarding claim 4, Denman discloses the apparatus of claim 1, wherein said early results comprise results of instructions comprising a subset of the instruction set supported by the microprocessor.

Note that a set is a subset of itself. The definition of subset according to WordNet ® 2.0, © 2003 Princeton University is a "set whose members are members of another set; a set contained within another set." Further, The American Heritage® Dictionary of the English Language, Fourth Edition Copyright © 2000 by Houghton Mifflin Company defines subset as "A set contained within a set." Using either of these *definitions, a set can be considered a subset of itself. Additionally, not all instructions use rename registers (e.g. non-operation instructions).*

16. Regarding claim 8, Denman discloses the apparatus of claim 4, wherein said subset of instructions comprises instructions used for updating the architected status flags for use as condition codes specified by conditional branch instructions (see col. 9, lines 29-33).

Note that branch instructions are based on conditions of previous instructions. These instructions update the architected status flags.

17. Regarding claim 9, Denman discloses the apparatus of claim 4, wherein said early results are generated prior to execution of the instructions (see col. 4, lines 26-28;

Note that the results are generated before the instructions are written to the architectural register file 16.)

by execution units

(Note that the definition of execution according to WordNet ® 2.0, © 2003 Princeton University is "the process of carrying out an instruction by a computer". Therefore, any unit for carrying out instructions is considered an execution unit. Therefore, the logic for writing results back (completing execution) is considered to be an execution unit.)

of the microprocessor that generate final results of the instructions.

(Note that the definition of generate according to The Free On-line Dictionary of Computing, © 1993-2005 Denis Howe is "To produce something according to an algorithm or program or set of rules, or as a (possibly unintended) side effect of the execution of an algorithm or program." Using this definition, results are generated by the sequencer (see col. 4, lines 29-30) when the results are moved from the rename buffer into the architectural register file.)

18. Regarding claim 10, Denman discloses the apparatus of claim 1, further comprising: early execution logic (Execution Unit 12; see fig. 1), coupled to said logic, for generating said early results of said preceding instructions (see col. 3, lines 32-36).

19. Regarding claim 11, Denman discloses the apparatus of claim 10, wherein said early results that modify said early status flags are valid if said instruction specifies an operation that said early execution logic is configured to perform and all input operands to said early execution logic used to generate said early results are valid (see col. 10, lines 12-24).

Note that results of an instruction are valid if all the inputs to that instruction are also valid. If a branch is mispredicted, the results in the rename buffer are invalidated; otherwise the results are valid.

20. Regarding claim 12, Denman discloses the apparatus of claim 10, wherein said early execution logic is within an address generation stage of the pipeline microprocessor (see col. 4, lines 50-56).

Note that the early execution logic is considered to be within an address generation stage because the address of where the result is to be stored in the rename buffer is calculated.

21. Regarding claim 13, Denman discloses the apparatus of claim 10, wherein said early results are generated immediately following operands corresponding to said early results being read from the architected register file (see col. 6, lines 25-45).

22. Regarding claim 14, Denman discloses the apparatus of claim 10, further comprising: an early register file, coupled to said early execution logic, having a plurality of registers corresponding to registers of an architected register file of the microprocessor (storage element; see above regarding claim 1), wherein said plurality of registers of said early register file are selectively valid (see col. 4, lines 63-67; col. 5, lines 1-11).

23. Regarding claim 15, Denman discloses the apparatus of claim 14, wherein if one of said plurality of registers provides an input operand to said early execution logic to generate said early results of at least one of said instructions that modify said early

status flags and said input operand is invalid, said logic invalidates said early status flags (see col. 10, lines 11-22;

Note that a branch instruction that is incorrectly predicted, it will invalidate the registers in the rename table that have been created because of the branch instruction. If an instruction is invalidated because of a branch misprediction, the operand in the rename table will be invalidated.).

24. Regarding claim 16, Denman discloses the apparatus of claim 10, wherein said early execution logic is configured to execute a subset of instructions executable by the microprocessor (all instructions; see above regarding claim 4), wherein if at least one of said instructions that modify said early status flags is not in said subset, said logic invalidates said early status flags.

Note that since this never happens (because all instructions are included in the subset), the claimed condition is always satisfied.

25. Regarding claim 17, Denman discloses the apparatus of claim 1, wherein the pipeline microprocessor is a scalar microprocessor (see col. 1, lines 22-30;

Note that the definition of scalar according to The Free On-line Dictionary of Computing, © 1993-2005 Denis Howe is "Any data type that stores a single value (e.g. a number or Boolean), as opposed to an aggregate data type that has many elements." Using this definition, the processor is considered to be scalar because it is not a vector processor.).

26. Regarding claim 18, Denman discloses the apparatus of claim 1, wherein the pipeline microprocessor issues instructions in program order (see col. 2, lines 5-22).

27. Regarding claim 19, Denman discloses the apparatus of claim 1, wherein said storage element is not specifiable by program instructions (see col. 5, lines 51-54).

Note that the reorder buffer allocates entries not specified by instructions, but by determining the first empty slot (see col. 7, lines 37-67).

28. Regarding claim 20, Denman discloses the apparatus of claim 1, wherein a computer program product comprising a computer usable medium having computer readable program code causes the apparatus (to operate), wherein said computer program product is for use with a computing device (see col. 2, lines 27-31).

Note that any instructions run on this processor cause the processor to operate.

29. Regarding claim 21, Denman discloses the apparatus of claim 1, wherein a computer data signal embodied in a transmission medium comprising computer-readable program code provides the apparatus (with data) (see col. 2, lines 27-31).

Note that any instructions run on this processor, provide the processor with data.

30. Regarding claim 22, Denman discloses a pipeline microprocessor having non-selectively valid architected status flags (registers in Architectural Register File 16; see

fig. 1; col. 34-38), and including in its instruction set conditional instructions (any instruction; see above regarding claim 1) that specify a condition (if necessary operands are available) and an operation (what the instruction performs), wherein if the condition is satisfied the microprocessor performs the operation (see col. 7, lines 15-25), comprising: early status flags (Results stored in Rename Buffer 14; see fig. 1) corresponding to the status flags stored in the architected register (see col. 4, lines 26-28), wherein said early status flags are selectively valid (see col. 4, lines 63-67; col. 5, lines 1-11); and early execution logic (Execution Unit 12), coupled to receive said early status flags (see col. 6, lines 49-52), for performing an operation specified by a conditional instruction (generating a result; see col. 3, lines 33-34) if said early status flags are valid (see col. 52-56), and if a condition specified by said conditional instruction (if necessary operands are available) is satisfied in said early status flags (see col. 7, lines 14-18).

Note that if the operands are not available, the execution unit will wait for the operands to become available.

31. Regarding claim 23, Denman discloses the microprocessor of claim 22, further comprising: final execution logic (Execution Unit 12), coupled to receive the architected status flags (see col. 6, lines 30-37), for performing said operation (generating a result; see col. 3, lines 33-34) if said condition is satisfied in the architected status flags (if the operand resides only in the architectural register file) and said operation is not performed by said early execution logic.

(Note that if the instruction were already executed, it would not need to be executed again. Therefore, the instruction is not executed by Execution Unit 12 before after it is already executed. Further note that Execution Unit 12 is considered to be the early execution logic and the final execution unit.)

32. Regarding claim 24, Denman discloses the microprocessor of claim 22, further comprising: logic, for generating said early status flags (Execution Unit 12) in response to an instruction preceding said conditional instruction (see col. 3, lines 32-36), wherein said preceding instruction specifies modification of the architected status flags (see col. 4, lines 26-28).

33. Regarding claim 25, Denman discloses the microprocessor of claim 24, further comprising: an early register file (Rename Buffer 14; see fig. 1), operatively coupled to said logic, having a plurality of registers corresponding to registers of an architected register file of the microprocessor (see col. 4, lines 26-28); wherein said plurality of registers of said early register file are selectively valid (see col. 4, lines 63-67; col. 5, lines 1-11).

34. Regarding claim 26, Denman discloses the microprocessor of claim 25, wherein said preceding instruction specifies at least one input operand from said architected register file, wherein said logic generates said early status flags in response to a result of said preceding instruction that is generated based on said at least one input operand

provided by said early register file rather than by said architected register file (see col. 6, lines 44-62 regarding obtaining an operand from the rename buffer).

35. Regarding claim 27, Denman discloses the microprocessor of claim 26, wherein if said at least one operand provided by said early register file is invalid, said early status flags are invalid (see col. 10, lines 11-14).

Note that if a branch is mispredicted and it causes the input operand of the instruction to become invalid the status flags would be invalidated.

36. Regarding claim 28, Denman discloses the microprocessor of claim 24, wherein if at least one input operand provided to said preceding instruction is invalid, said early status flags are invalid (see col. 10, lines 11-14).

Note that if a branch is mispredicted and it causes the input operand of the instruction to become invalid the status flags would be invalidated.

37. Regarding claim 29, Denman discloses the microprocessor of claim 24, wherein said early execution logic is coupled to said logic, and is configured to execute a subset of instructions executable by the microprocessor (see above regarding claim 4), wherein if said preceding instruction is not in said subset, said early status flags are invalid (see above regarding claim 16).

38. Claim 30 recites equivalent limitations as claim 17 and is therefore rejected under the same grounds.

39. Claim 31 recites equivalent limitations as claim 18 and is therefore rejected under the same grounds.

40. Claim 32 recites equivalent limitations as claim 12 and is therefore rejected under the same grounds.

41. Regarding claim 33, Denman discloses a method for generating status flags early in a pipeline microprocessor to enable early execution of an instruction (execution before data is available from architected registers) that conditionally performs an operation based on a condition of the status flags (if necessary operands are available) specified by the instruction, the method comprising: generating a first instance of the status flags in response to an instruction preceding the conditional instruction (see col. 3, lines 32-36) wherein the first instance may not be valid (see col. 4, lines 63-67; col. 5, lines 1-11); generating a second instance of the status flags in response to the preceding instruction (see above regarding claim 9), subsequent to said generating the first instance (see above regarding claim 2), wherein the second instance is always valid (see col. 6, lines 36-39); and performing the operation, prior to said generating the second instance (the instruction must be executed before it can be written back), if the

condition is satisfied in the first instance of the status flags (if necessary operands are available; see col. 7, lines 14-18;

Note that if the operands are not available, the execution unit will wait for the operands to become available.)

and if the first instance is valid (see col. 52-56).

42. Regarding claim 34, Denman discloses the method of claim 33, further comprising: updating an architected register for storing the status flags after said generating said second instance (see col. 3, lines 38-42).

43. Regarding claim 35, Denman discloses the method of claim 33, further comprising: generating a result of the preceding instruction, prior to said generating the first instance (see col. 7, lines 14-24;

Note that the results of a previous instruction must be known before the instruction that needs the result can be executed.);

determining whether the result of the preceding instruction is valid; and invalidating the first instance if the result is invalid (see col. 10, lines 12-22).

44. Regarding claim 36, Denman discloses the method of claim 35, wherein said determining whether the result of the preceding instruction is valid comprises: determining whether the result of the preceding instruction is generated using valid input operands; and indicating the result is invalid if the input operands are invalid.

Note that if a branch is mispredicted, the inputs of instructions as well as the results are invalid.

45. Regarding claim 37, Denman discloses the method of claim 35, wherein said determining whether the result of the preceding instruction is valid comprises: determining whether the preceding instruction is an instruction that is performable by early execution logic of the microprocessor; and indicating the result is invalid if the preceding instruction is not an instruction that is performable by the early execution logic (see col. 10, lines 12-22).

Note that the branch prediction logic is considered to be the early execution logic. Therefore, if a branch is mispredicted, it was not correctly performed by the early execution logic.

46. Regarding claim 38, Denman discloses the method of claim 37, wherein the early execution logic generates the result prior to generation of an always valid instance of the result by an execution unit of the microprocessor (see col. 8, lines 1-19).

Note that a branch predictor works because it gives speculative results prior to always valid results.

47. Regarding claim 39, Denman discloses the method of claim 35, further comprising: determining whether the preceding instruction modifies the status flags; and

said invalidating the first instance only if the preceding instruction modifies the status flags (see col. 6, lines 45-62; col. 4, lines 65-68).

Note that in the rename buffer, only entries corresponding to invalid instructions are invalidated.

48. Regarding claim 40, Denman discloses the method of claim 33, wherein said generating the first instance is performed without stalling the microprocessor pipeline regardless of whether input operands to the preceding instruction are valid (see col. 8, lines 1-19).

Note that when an instruction is determined to be invalid, the processor is not stalled, but the pipeline is flushed.

49. Note that claim 43 recites equivalent limitations as claim 1 and is therefore rejected under the same grounds. Note that the code for providing the processor data is all instructions executed on the processor (see above regarding claim 21).

Claim Rejections - 35 USC § 103

50. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

51. Claims 3, and 5-7 are rejected under 35 U.S.C. 103(a) as being unpatentable over Denman in view of Intel (Intel Architecture Software Developer's Manual Volume 2: Instruction Set Reference).

52. Regarding claim 3, Denman discloses the apparatus of claim 1.

Denman does not expressly disclose said early status flags comprise one or more x86 architecture EFLAGS register status flags.

Intel teaches said early status flags comprise one or more x86 architecture EFLAGS register status flags (see pages 2-7, 2-8 and 2-9; section 2.3).

It would have been obvious at the time of the invention for one of ordinary skill in the art to have modified the invention of Denman by including one or more x86 architecture EFLAGS register status flags in said early status flags, as taught by Intel, in order to control I/O, maskable hardware interrupts, debugging, task switching, and the virtual-8086 mode (see last 3 lines of page 2-7).

53. Regarding claim 5, Denman discloses the apparatus of claim 4.

Denman does not expressly disclose said subset of instructions comprises instructions for performing simple arithmetic operations.

Intel teaches said subset of instructions comprises instructions for performing simple arithmetic operations (see page 15-4; second last bullet).

It would have been obvious at the time of the invention for one of ordinary skill in the art to have modified the invention of Denman by including instructions for performing simple arithmetic operations in said subset of instructions, as taught by Intel, in order to perform arithmetic operations on data. It is extremely well known in the art that performing arithmetic instructions on data is useful.

54. Regarding claim 6, Denman discloses the apparatus of claim 4.

Denman does not expressly disclose said subset of instructions comprises instructions for performing simple shift operations.

Intel teaches said subset of instructions comprises instructions for performing simple shift operations (see page 15-5; fourth bullet).

It would have been obvious at the time of the invention for one of ordinary skill in the art to have modified the invention of Denman by including instructions for performing simple shift operations in said subset of instructions, as taught by Intel, in order to perform shift operations on data. It is extremely well known in the art that performing shift instructions on data is useful.

55. Regarding claim 7, Denman discloses the apparatus of claim 4.

Denman does not expressly disclose said subset of instructions comprises instructions for performing simple Boolean operations.

Intel teaches said subset of instructions comprises instructions for performing simple Boolean operations (see page 15-4; last bullet).

It would have been obvious at the time of the invention for one of ordinary skill in the art to have modified the invention of Denman by including instructions for performing simple Boolean operations in said subset of instructions, as taught by Intel, in order to perform Boolean operations on data. It is extremely well known in the art that performing Boolean instructions on data is useful.

Allowable Subject Matter

56. Claims 41 and 42 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

57. The following is a statement of reasons for the indication of allowable subject matter:

58. Claim 41 recites the limitation "determining whether the microprocessor pipeline is flushed; and copying architected status flags to the first instance of the status flags and validating the first instance of the status flags if the microprocessor pipeline is flushed." The prior art does not teach or fairly suggest updating temporary status flags (as recited in claim 1) by copying architected status flags when the microprocessor pipeline is flushed.

59. Claim 42 recites the limitation "determining whether all status flag-modifying instructions present in the microprocessor pipeline below a stage in which said generating the first instance is performed, if any, have updated architected status flags

of the microprocessor; and copying architected status flags to the first instance of the status flags and marking the first instance of the status flags valid if all status flags-modifying instructions present in the microprocessor pipeline below a stage in which said generating the first instance is performed, if any, have updated architected status flags of the microprocessor." The prior art does not teach or fairly suggest updating temporary status flags (as recited in claim 1) by copying architected status flags when all status flags modifying instructions present in the pipeline below a stage in which said generating the first instance is performed have updated architected status flags.

Conclusion

60. The following is text cited from 37 CFR 1.11(c): In amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jesse R. Moll whose telephone number is (571)272-2703. The examiner can normally be reached on M-F 8:00 am - 4:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Fritz M. Fleming can be reached on 571-272-4145. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

JM 4/2/2006

Supervisory
FRITZ FLEMING
PRIMARY EXAMINER
GROUP 2100
AU2181
4/2/2006